# Assessment type (🗹)

|  |  |
| --- | --- |
|  | Questioning (Oral/Written) |
|  | Practical Demonstration |
|  | 3rd Party Report |
|  | Other – Project/Portfolio (*please specify below)* |
|  | Part 3 of a portfolio of practical coding and electronics tasks |

# Student Details

Please complete this table with your details:

|  |  |  |
| --- | --- | --- |
| Given Name | Family Name | Student ID |
|  |  |  |

# Version Details

|  |  |  |  |
| --- | --- | --- | --- |
| V | Date | Editor | Summary |
| 1.3 | 2025-05-07 | A Gould | Changed code to a new structure, and use a base FastAPI application to facilitate easier completion of the assessment. |
| 1.2 | 2025-04-02 | A Gould | Add section for basic authentication that verifies user/password |
| 1.1 | 2024-08-28 | A Gould | Minor updates to fix errors in instructions |
| 1 | 2024-07-12 | A Gould | New version using updated template |

**Note:** On the following page is a table of contents to assist you to navigate this document. You may CTRL+CLICK on an entry to jump to that location.

# Parts of the Document

**Assessment Instructions** These are the instructions that must be followed whilst completing the assessment.

**Assessment Instrument** This is where you may be asked questions, required to supply evidence of your work and other specific information as required.

**Appendices** Further required information that are required guidelines, but are better placed external to the work to be performed.

# Table of Contents

[Assessment type (🗹) 1](#_Toc197530990)

[Student Details 1](#_Toc197530991)

[Version Details 1](#_Toc197530992)

[Parts of the Document 2](#_Toc197530993)

[Table of Contents 2](#_Toc197530994)

[Assessment Resources 4](#_Toc197530995)

[College to Provide 4](#_Toc197530996)

[Student to Provide 4](#_Toc197530997)

[Assessment Instructions 5](#_Toc197530998)

[Date Due 5](#_Toc197530999)

[Scenario 5](#_Toc197531000)

[Information Referencing 6](#_Toc197531001)

[Before Commencing 6](#_Toc197531002)

[Step 1: Setting Up 7](#_Toc197531003)

[Step 2: Starter Application 13](#_Toc197531004)

[Step 3: Policy Based Access Control 15](#_Toc197531005)

[Step 4: Middleware 18](#_Toc197531006)

[Step 5: Implement Middleware 20](#_Toc197531007)

[Step 6: Authenticating a User 22](#_Toc197531008)

[Step 7: Bob the User… Can They Access It? 24](#_Toc197531009)

[Step 8: Password and Username Validation 26](#_Toc197531010)

[Step 8: Submission 30](#_Toc197531011)

[Assessment Instrument 31](#_Toc197531012)

[Evidence 1 Setting Up 31](#_Toc197531013)

[Evidence 2 Basic API Docs 31](#_Toc197531014)

[Reflection Question 1 32](#_Toc197531015)

[Reflection Question 2 33](#_Toc197531016)

[Reflection Question 3 33](#_Toc197531017)

[Reflection Question 4 33](#_Toc197531018)

[Evidence 3 Implement Middleware 34](#_Toc197531019)

[Reflection Question 5 34](#_Toc197531020)

[Reflection Question 6 35](#_Toc197531021)

[Reflection Question 7 35](#_Toc197531022)

[Evidence 4 User Access 36](#_Toc197531023)

[Reflection Question 8 36](#_Toc197531024)

[Evidence 5 User Access 2 36](#_Toc197531025)

[Reflection Question 9 37](#_Toc197531026)

[Reflection Question 10 37](#_Toc197531027)

[Evidence 6 Test Results 38](#_Toc197531028)

[Appendix A: Assessment Submission 39](#_Toc197531029)

[Appendix B: Code Style and Commenting 40](#_Toc197531030)

[Appendix C: Referencing 44](#_Toc197531031)

[Appendix D: Video Recordings 45](#_Toc197531032)

[Appendix E: File Naming 46](#_Toc197531033)

[Arduino Projects 46](#_Toc197531034)

[Git Ignore File 46](#_Toc197531035)

# Assessment Resources

|  |  |
| --- | --- |
| College to Provide  * Web server, Python interpreter and database server * Access to the Internet (both Ethernet and Wi-Fi based) * Access to an MQTT broker * IDE or editor for developing Python server/desktop/embedded programs (only PyCharm supported by the college) * IDE or editor for developing C/C++ embedded programs (Arduino IDE v2+ or CLion) * Arduino Uno, ESP32, Raspberry Pi Pico W, or Raspberry Pi with various sensors and actuators * Access to Office 365 & Microsoft Word * TinkerCAD Account (created by assessor/lecturer – if you do not have a classroom account, please use ScreenCraft Helpdesk to make the request) | Student to Provide Students may optionally obtain their own Electronics Kit to use with this and all other assessment items in this cluster.  Online Students must provide their own electronics kit for use with this cluster.  Details are shown in the Blackboard shell. |

# Assessment Instructions

* Please read these instructions carefully.
* Follow each step as provided.
* Questions will be asked in a separate section of this document, with space provided for your answers.
* Information in the appendices **MUST** be applied to your assessment submissions.

## Date Due

* 5:30PM on day of Session 14

## Scenario

You are employed as a junior embedded systems developer for RIoT Systems (Robotics & Internet of Things), a Perth based educational and development company who specialise in IoT and Robotics systems.

You have been tasked with the implementation of a Wirelessly connected IoT Device and it’s controlling software.

You are provided with a set of requirements to accomplish this (this document).

At any stage during this assignment item, you may consult the stakeholder(s) or their representative(s).

## Information Referencing

This is COMPULSORY for all assessments and covers resources that include but is not limited to:

* The Internet;
* Books;
* Video;
* Code;
* AI Use; and
* Audio.

More details on referencing requirements may be found in Appendix C: Referencing.

## Before Commencing

### Familiarisation

Familiarise yourself with the content of this assessment by reading the whole document **at least once** before commencing.

As you progress through the steps contained in this assessment document, any questions relating to a step, or required evidence will be added into the Assessment Instrument section.

### Versioning, Bash Shell and MS Terminal

* For details on setting up Windows Terminal with the Git Bash CLI see [Add Git Bash to Microsoft Terminal](https://help.screencraft.net.au/hc/2680392001/65/add-git-bash-to-microsoft-terminal?category_id=35) (CTRL+CLICK to open link).
* For details on adding useful aliases to the terminal see [Add or Update Bash Command Line Aliases for Git, MailPit and more](https://help.screencraft.net.au/hc/2680392001/66/add-bash-command-line-aliases-for-git?category_id=35) (CTRL+CLICK to open link).

### Useful Resources for this Assessment

Please see the following links for useful references for this assessment:

* <https://www.diigo.com/user/ady_gould>

It is important you check these resources as they contain very useful resources for this and other areas of interest.

## Step 1: Setting Up

We are first going to set up our project.

### Set up Folders and Basic Files

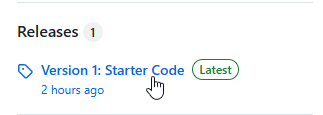
To make things a little easier, we provide a web site based on FastAPI to use as the basis of this assessment item.

You will need to modify the structure whilst completing this assessment item.

Open the following URI: https://github.com/AdyGCode/xxx-fastapi-base-code

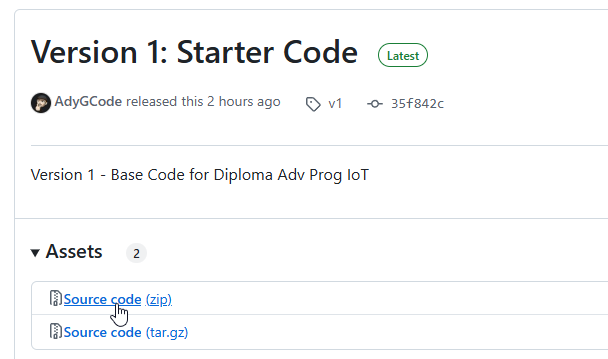
Locate the Releases section of the page, and locate the latest release.

Click this link



Example (Version 1): https://github.com/AdyGCode/xxx-fastapi-site-base/releases/tag/v1

You will then download the Source Code from this page:



Once the file is downloaded, you will need to uncompress it. We strongly suggest using the 7-Zip application to do this as it is many times faster than the MS Windows built in compression engine.

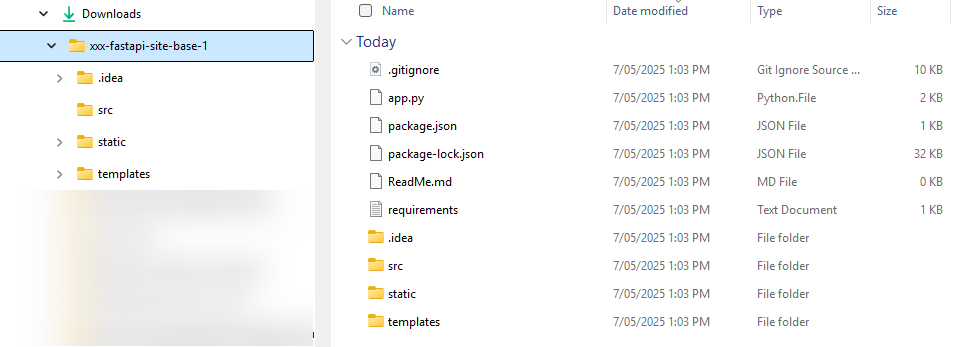
Open the Windows File Explorer window using WIN+E

Go to your Downloads folder

Right Mouse on the zip file, and select the “Other Options” from the pop up menu.

Within this select “7-Zip” and then the “Extract to xxx-fastapi-site-base-1” option.

If you double click the new folder, you should see something similar to this:



We now want to move these files to our Source/Repos folder.

Open the Microsoft Terminal making sure it uses the Bash CLI.

Execute the following to move the folders and files and rename the folder to the desired name for this assessment, remembering to **REPLACE** XXX **WITH YOUR INITIALS**.

cd ~/Source/Repos

mv xxx-fastapi-site-base-1 XXX-InterRIoT-AT2-POR-Pt3-2025-S1

cd XXX-InterRIoT-AT2-POR-Pt3-2025-S1

Now we create a new Python Virtual environment and activate it for use.

python -m venv .venv

source .venv/Scripts/activate

(Mac users will use “bin” in place of Scripts)

Verify Python 3.12+ is being used, and where it is being executed from:

python -V

which python

The output from these will be similar to:

Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)]

/c/Users/USERNAME/Source/Repos/XXX-InterRIoT-AT2-Pt3-YYY-SN/.venv/Scripts/python

Next create the folder structure and placeholder ignore files for the project using the CLI:

mkdir -p {app,src,templates,static,api}

touch {app,src,templates,static,api}/.gitignore

touch requirements.txt

You are provided with a “.gitignore” file, so no need to re-create it.

We also provide, with this set of files, a “ReadMe.md” file.

### Configure and Initialise Version Control

As a precaution, set git defaults (replace the YOUR NAME etc with the relevant details).

git config --global user.name “YOUR NAME”

git config --global user.email “YOUR EMAIL ADDRESS”

git config --global init.defaultBranch main

Initialise a git repository:

git init .

### Install Packages

From the command line we will install the following packages:

|  |  |
| --- | --- |
| Package | Minimum Version |
| casbin | 1.41.0 |
| fastapi | 0.115.12 |
| fastapi-authz | 1.0.0 |
| setuptools | 78.1.0 |
| uvicorn | 0.34.0 |

A set of requirements is already configured as part of the code you downloaded. This is the requirements.txt file.

WARNING: Before executing the commands that follow, ensure that you have got your Python virtual environment activated using the source .venv/Scripts/activate command.

We will install these requirements by running the following commands in the current virtual environment:

python.exe -m pip install --upgrade pip

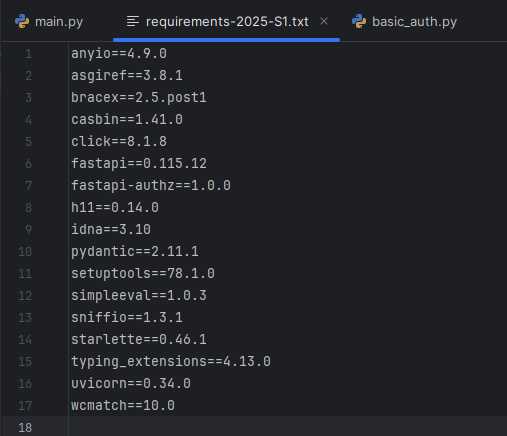
pip install -r requirements.txt

pip install --upgrade setuptools uvicorn fastapi-authz fastapi casbin

Once complete we will “freeze” these requirements using:

python -m pip freeze > requirements.txt

Make sure the requirements file looks similar to this:



Yours will be different from the above.

### Update Version Controlled Files

Check version control status:

git status

Add the requirements and git ignore files to version control:

git add .

Commit the files

git commit -m “init: Start of Portfolio Part 3, PBAC”

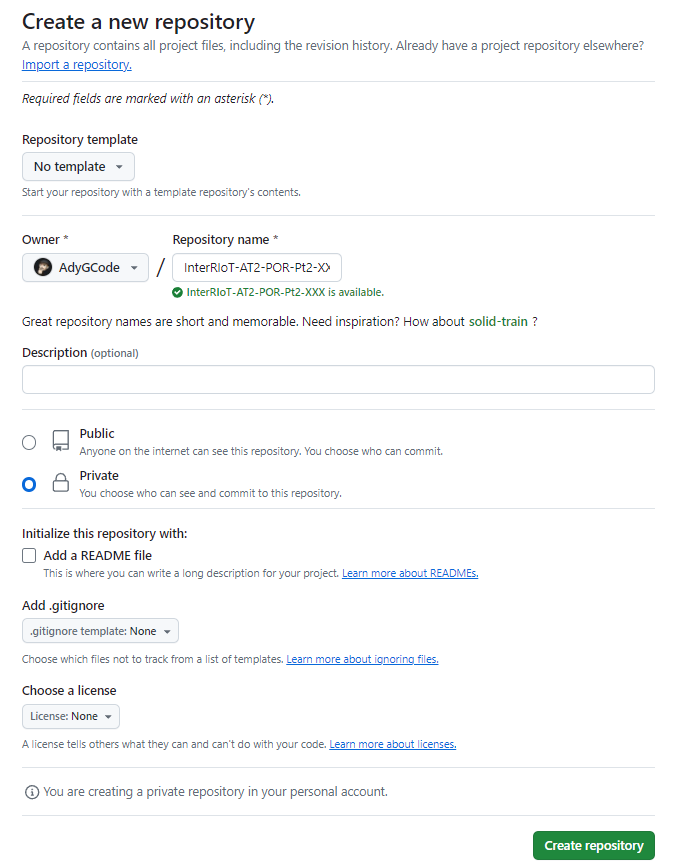
Take a screenshot of the result of your commit action and add to Evidence 1 Setting Up.

### Adding To GitHub

Open your GitHub account.

Create a new repository with the following details:

* Name: XXX-InterRIoT-AT2-POR-Pt3-2025-S1 (replace XXX with your initials)
* Files: NIL!
* Private: YES!



Add the remote to your local repository (single line):

git remote add origin https://github.com/USERNAME/XXX-InterRIoT-AT2-POR-Pt3-2025-S1.git

Push the changes to the remote

git push -u origin main

## Step 2: Starter Application

We will now create the basic API Application.

Open PyCharm and open the project folder (XXX-InterRIoT-AT2-POR-Pt3-2025-S1).

### Starter Application

Copy the base\_code.py file from the src folder to the root of the project, renaming it api.py.

cp src/base\_code.py api.py

Open this file, and update the foundation by editing the code as instructed below.

Make sure that the following line is in the file:

from fastapi import FastAPI

Change the app = FastAPI line to read:

api = FastApi(‘XXX-InterRIoT-AT2-POR-Pt3-2025-S1’)

After the “/” route and index function, add:

@api.get(‘/api’)

async def api\_index():

return ‘Hello, world.’

Open a terminal in PyCharm and run the following command from the project’s ‘root’ folder:

.venv/Scripts/fastapi dev api.py --reload

Open the link <http://127.0.0.1:8000/docs> in your browser, and read the information that is presented on this page.

**Try out** the newly created endpoint (/api) via the web docs and add a screenshot of the endpoint test in Evidence 2 Basic Api Docs.

### Accessing the endpoint via CLI

Split your terminal in half using ALT+SHIFT+-

Click in the bottom section and enter the following command:

curl -X ‘GET’ ‘http://127.0.0.1:8000/api’ -H ‘accept: application/json’

What are the results?

Take a screenshot and add to Screenshot of API results from CLI.

A black screen with white numbers

AI-generated content may be incorrect.

What did the FastAPI command show?

Take a screenshot and add to Screenshot of FastAPI Output on CLI.

A screenshot of a computer program

AI-generated content may be incorrect.

### Add to Version Control

Use the following to add to version control:

git add .

Commit and push the files:

git commit -m “feat(step-2): Create basic FastAPI application”

git push origin main

## Step 3: Policy Based Access Control

Now we will add the PBAC.

* PBAC – Policy Based Access Control
* RBAC - Role-based Access Control

We will implement RBAC, which is a subset of PBAC.

### Create the policy files

To enable PBAC (RBAC), we will first create two files in the api/ folder:

* rbac\_model.conf and
* rbac\_policy.csv.

Use the following CLI command:

touch api/{rbac\_model.conf,rbac\_policy.csv}

Add the following lines to the files:

#### rbac\_model.conf

[request\_definition]

r = sub, obj, act

[policy\_definition]

p = sub, obj, act

[role\_definition]

g = \_, \_

[policy\_effect]

e = some(where (p.eft == allow))

[matchers]

m = (p.sub == “\*” || g(r.sub, p.sub)) && (r.obj == p.obj || keyMatch(r.obj, p.obj)) && (p.act == “\*” || r.act == p.act)

The last line is just a single line. Make sure your spacing and punctuation are correct.

#### [rbac\_policy.csv]

p, alice, /api/ds1/\*, GET

p, alice, /api/ds1/res1, POST

p, bob, /api/ds2/res2, GET

### Activate the Policies in Code

Add the following lines to your main script (highlighted):

import casbin

from fastapi import FastAPI

api = FastApi()

enforcer = casbin.Enforcer(

‘api/rbac\_model.conf’,

‘api/rbac\_policy.csv’,

)

@api.get(‘/api’)

async def api\_index():

return ‘Hello, world.’

Monitor the output of the FastAPI process in your terminal to check for errors (it will try to restart because of --reload).

Even though the newly create enforcer isn’t used, it should load the required files correctly.

Fix any errors you encounter.

### Add to Version Control

Use the following to add to version control:

git add .

Commit and push the files:

git commit -m “feat(step-3): Add PBAC policies and models”

git push origin main

### Reflection Questions

#### Answer the Reflection Question 1, Reflection Question 2 and Reflection Question 3.

## Step 4: Middleware

### Create the required middleware

You will now add the required middleware. The package that you are using integrates nicely with any middleware that implement the AuthenticationMiddleware interface.

Create a new file called basic\_auth.py. It should reside in the / folder.

Add the following code to the file:

import base64

import binascii

from starlette.authentication import (

AuthenticationBackend,

AuthenticationError,

AuthCredentials,

SimpleUser

)

class BasicAuth(AuthenticationBackend):

async def authenticate(self, request):

if ‘Authorization’ not in request.headers:

return None

auth = request.headers[‘Authorization’]

try:

scheme, credentials = auth.split()

decoded = base64.b64decode(credentials).decode(‘ascii’)

except (ValueError, UnicodeDecodeError, binascii.Error):

raise AuthenticationError(‘Invalid credentials’)

username, \_, password = decoded.partition(‘:’)

return AuthCredentials([‘authenticated’]), SimpleUser(username)

As you can see, this class implements a method called authenticate.

### Reflection Questions

#### Answer the Reflection Question 4.

### Adding Updated/New Files to Versioning

Use the following to add to version control:

git add .

Commit and push the files:

git commit -m “feat(step-4): Adding PBAC Middleware”

git push origin main

## Step 5: Implement Middleware

For the access control to work properly, we need to add the appropriate middleware to the API. Follow these instructions…

Update your main script (api.py) by adding the following lines (highlighted):

import casbin

from fastapi import FastAPI

**from fastapi\_authz import CasbinMiddleware**

**from starlette.middleware.authentication import AuthenticationMiddleware**

**import api.basic\_auth as basic\_auth**

api = FastApi()

enforcer = casbin.Enforcer(

‘rbac\_model.conf’,

‘rbac\_policy.csv’,

)

**backend = basic\_auth.BasicAuth()**

**api.add\_middleware(CasbinMiddleware, enforcer=enforcer)**

**api.add\_middleware(AuthenticationMiddleware, backend=backend)**

@api.get(‘/api’)

async def api\_index():

**return ‘{“success”:true, “message”: “Welcome”, “data”: [“greeting”: “Hello, world.”]}’**

Open the following link in your browser:

* [**http://127.0.0.1:8000/docs/**](http://127.0.0.1:8000/docs/). (It’s the same link as before.)

Add a screenshot of the results in **Evidence 3 Implement Middleware**.

### Reflection Question

Complete the **Reflection Question 5** before continuing.

### Update Policies

Update the policy file to provide GET access for a user anonymous on the endpoints /docs and /openapi.json.

In the CLI, use CTRL+C to stop the FastAPI server.

Restart the FastAPI server manually (.venv/Scripts/fastapi dev api.py --reload), and try loading the page again.

### Reflection Questions

Complete the **Reflection Question 6** and **Reflection Question 7** before continuing.

### Adding Updated/New Files to Versioning

Use the following to add to version control:

git add .

Commit and push the files:

git commit -m “feat(step-5): Implement Middleware”

git push origin main

## Step 6: Authenticating a User

### Authenticate a user

In **Step 5: Implement Middleware**, we have opened the API for abuse by anonymous users by allowing access to certain endpoints for non-authenticated users. You would not do this in a production system!

In this step, we will try to access the endpoint /api for an authenticated user, Alice.

You can either use the curl tool from the command line (Linux/MAC) or something like Postman (all OS) to add credentials to the request.

#### Testing with a user

In a terminal, run the following command:

curl -i -u alice:password <http://127.0.0.1:8000/>api

Provide a screenshot of the results in **Evidence 4 User Access** and answer the **Reflection Question 8**.

#### Update Policy

Add the following line to the policy file:

p, alice, /api, GET

Restart the FastAPI server manually (it only watches for changes in the main script).

Run the curl command again.

Describe what you see this time and provide a screenshot in **Evidence 5 User Access 2** and **Reflection Question 9**.

### Adding Updated/New Files to Versioning

Use the following to add to version control:

git add .

Commit and push the files:

git commit -m “feat(step-6): Authenticating Jane”

git push origin main

## Step 7: Bob the User… Can They Access It?

### Adding endpoints for Bob too

Your policy file should now look like this (some lines have been omitted):

p, alice, /api, GET

p, alice, /api/ds1/\*, GET

p, alice, /api/ds1/res1, POST

p, bob, /api/ds2/res2, GET

Now, follow these instructions…

Add two new endpoints (follow the existing code for the endpoint /api as guidance):

* /api/ds1/res1 and
* /api/ds2/res2.

The endpoints can just return a string.

Make sure the server restarts without problems after making any changes.

Describe the output of testing the endpoint with the command below in **Reflection Question 10**.

curl -i -u alice:password <http://127.0.0.1:8000/>api

Content continues on next page…

Test the URLs (GET using curl or Postman) and record the results in the table provided in **Evidence 6 Test Results**.

|  |  |
| --- | --- |
| Endpoint | User |
| /api/ds1/res1 | anonymous |
| /api/ds2/res2 | anonymous |
| /api/ds1/res1 | alice |
| /api/ds2/res2 | alice |
| /api/ds1/res1 | bob |
| /api/ds2/res2 | bob |

When testing, if the user is not anonymous, assume the password is password.

### Adding Updated/New Files to Versioning

Use the following to add to version control:

git add .

Commit and push the files:

git commit -m “feat(step-7): Authenticating Bob”

git push origin main

## Step 8: Password and Username Validation

Currently the code does not check to see if the user and the password are correct.

We will now “hard code” a password validation to the mini-application.

### Update the Basic Auth

Open the Basic Auth file and update it as shown below.

#### Update the imports as needed

import base64

import binascii

from starlette.authentication import (

AuthenticationBackend,

AuthenticationError,

AuthCredentials,

SimpleUser

)

The first part of the Basic Auth Class should be unchanged…

class BasicAuth(AuthenticationBackend):

async def authenticate(self, request):

if ‘Authorization’ not in request.headers:

return None

auth = request.headers[‘Authorization’]

Our first change is in returning an Authentication Error to indicate we have an invalid authentication scheme…

try:

scheme, credentials = auth.split()

if scheme.lower() != “basic”:

raise AuthenticationError(“Invalid authentication scheme”)

decoded = base64.b64decode(credentials).decode(‘ascii’)

except (ValueError, UnicodeDecodeError, binascii.Error):

raise AuthenticationError(“Invalid credentials format”)

username, \_, password = decoded.partition(‘:’)

# Simple hardcoded check (replace with database or hashed password verification)

if username not in [“alice”, “bob”] or password != “password”:

raise AuthenticationError(“Invalid credentials”)

return AuthCredentials([“authenticated”]), SimpleUser(username)

#### Update the Main API program file

Update the imports to include:

from starlette.authentication import requires, AuthenticationError

from fastapi.responses import HTMLResponse, JSONResponse

Update the code by adding the following IMMEDIATELY before the @api.get(‘/api’) line.

@api.exception\_handler(AuthenticationError)

async def auth\_exception\_handler(request: Request, exc: AuthenticationError):

*“““Custom authentication error handler to return JSON response”““*

return JSONResponse(

content={“success”: False, “message”: str(exc), “data”: []},

status\_code=401

)

Update the return from the api\_index method to ensure it is a JSON Response:

return JSONResponse(content={

“success”: True,

“message”: “Endpoint: /”,

“data”: [

{“greeting”: “Hello, world.”}

]

})

Update the remaining response methods so they also respond with code that is similar to the /api endpoint. For example /api/ds1/res1’s response could become:

return JSONResponse(content={

“success”: True,

“message”: “Endpoint: ds1/res1”,

“data”: [

{“greeting”: “Resource 1 accessed”}

]

})

Likewise, for /api/ds2/res2:

return JSONResponse(content={

“success”: True,

“message”: “Endpoint: ds2/res2”,

“data”: [

{“greeting”: “Resource 2 accessed”}

]

})

Content continues on next page…

Test the URLs (GET using curl or Postman) and record the results in the table provided in **Evidence 6 Test Results**.

|  |  |
| --- | --- |
| Endpoint | User |
| /api/ds1/res1 | anonymous |
| /api/ds2/res2 | anonymous |
| /api/ds1/res1 | alice |
| /api/ds2/res2 | alice |
| /api/ds1/res1 | bob |
| /api/ds2/res2 | bob |

When testing, if the user is not anonymous, assume the password is password.

Remember the curl command will look similar to this:

curl -i -u alice:password <http://127.0.0.1:8000/>api

### Adding Updated/New Files to Versioning

Use the following to add to version control:

git add .

Commit and push the files:

git commit -m “feat(step-7): Authenticating Bob”

git push origin main

## Step 9: Catch All

### Handling Endpoints that may not exist

The last item is to handle endpoints that may be missing.

This is done using an ‘exception handler’.

Add the following to the end of your application code:

@api.exception\_handler(AuthenticationError)  
async def auth\_exception\_handler(request: Request, exc: AuthenticationError):  
 *"""Custom authentication error handler to return JSON response"""* return JSONResponse(  
 content={"success": False, "message": str(exc), "data": []},  
 status\_code=401  
 )

Try using the curl command to access some unknown endpoints:

|  |  |  |
| --- | --- | --- |
| Endpoint | User | Password |
| /api/ds1/ | alice |  |
| /api/health | anonymous |  |

## Step 10: Submission

Make sure you follow the details in Appendix A: Assessment Submission and Answer 11 Submission Requirements when submitting your assessment.

For this assessment we require:

* This document with all questions answered, all required screenshots and any code that has been requested to be copied and pasted into the document.
* A compressed copy of the Python project code **WITHOUT** the .venv folder.
* A copy of the video evidence recording if required.

# Assessment Instrument

When a step includes a question, you must attempt to answer it.

There is a minimum and maximum number of words to use for each answer.

If a step has more than one question, these maxima and minima are a total for all the questions in that specific step.

All answers must be in complete sentences unless indicated.

Unless otherwise directed, make sure to add any code you’ve written in a separate file to your submission. Also, unless otherwise directed, DO NOT put code in a Word document.

## Evidence 1 Setting Up

### Screenshot

Delete the example shown below and insert a screenshot of the commit results.

A screen shot of a computer program

AI-generated content may be incorrect.

## Evidence 2 Basic API Docs

### Screenshot of API Docs in Browser

Delete the example shown below and insert a screenshot of the API Docs.

A screenshot of a computer

AI-generated content may be incorrect.

### Screenshot of API results from CLI

Delete the example shown below and insert a screenshot of the curl command.



### Screenshot of FastAPI Output on CLI

Delete the example shown below and insert a screenshot of the FastAPI output.



## Reflection Question 1

#### Describe the meaning of “sub”, “obj” and “act” in the configuration file. [3 – 10 sentences]

Sub: means “subject”, the type of user the config applies to.

Obj: means “object”, the resource the subjects has or doesn’t have access to.

Act: means “action”, the method used. E.g. GET

## Reflection Question 2

#### In your own words, describe the authorisations that Alice and Bob have in the policy file. [3 – 5 sentences]

## Reflection Question 3

#### How do “sub”, “obj” and “act” relate to the various fields in the policy file? [3 – 10 sentences]

## Reflection Question 4

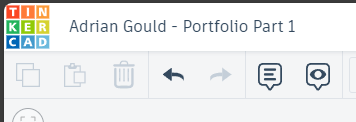
You may refer to any documentation that may exist for the class AuthenticationBackend.

#### In your own words describe what the authenticate method does. [3 – 10 sentences]

## Evidence 3 Implement Middleware

### Screenshot

Delete the example shown below and insert a screenshot visiting the URI.



## Reflection Question 5

#### Describe what you saw when visiting the URI [**http://127.0.0.1:8000/docs/**](http://127.0.0.1:8000/docs/) **after** making the changes to the main.py file. [3 - 10 sentences]

## Reflection Question 6

#### In your own words, describe what you saw when visiting the URI [**http://127.0.0.1:8000/docs/**](http://127.0.0.1:8000/docs/) **after** making the changes to the **policy** file and restarting FastAPI. [1 - 5 sentences]

#### In your own words, describe why this happened. [3 - 10 sentences]

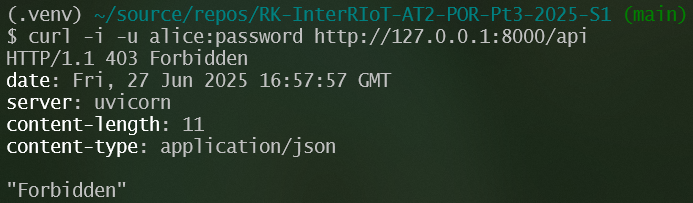
## Reflection Question 7

#### In your own words, describe what happens when you visit [**http://127.0.0.1:8000/**](http://127.0.0.1:8000/)? [1 – 5 sentences]

## Evidence 4 User Access

### Screenshot

Delete the example shown below and insert a screenshot visiting the URI.



## Reflection Question 8

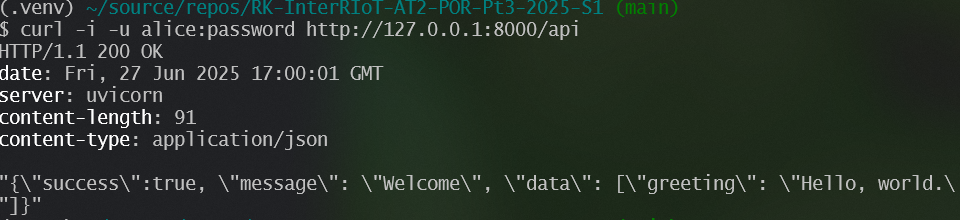
#### Describe the results of using the curl command (or postman) to visit <http://127.0.0.1:8000> with the user alice and their password. [1 - 10 sentences]

curl -i -u alice:password <http://127.0.0.1:8000/>

## Evidence 5 User Access 2

### Screenshot

Delete the example shown below and insert a screenshot visiting the URI.



## Reflection Question 9

#### After updating the policy file, and restarting FastAPI server, what is the result of the same curl command?

curl -i -u alice:password <http://127.0.0.1:8000/>

#### Describe the output in your own words [1-5 sentences].

## Reflection Question 10

#### After updating the policy file, and restarting FastAPI server, what is the result of the same curl command?

curl -i -u alice:password <http://127.0.0.1:8000/>

#### Describe the output in your own words [1-5 sentences].

## Evidence 6 Test Results

#### Complete the following table with the results of your testing:

|  |  |  |
| --- | --- | --- |
| Endpoint | User | Result |
| /api/ds1/res1 | anonymous |  |
| /api/ds2/res2 | anonymous |  |
| /api/ds1/res1 | alice |  |
| /api/ds2/res2 | alice |  |
| /api/ds1/res1 | bob |  |
| /api/ds2/res2 | bob |  |

# Appendix A: Assessment Submission

These assessment submission guidelines are common for all submissions in this cluster.

**DO NOT COMPRESS** any of the following when submitting:

* MS Office Documents (including Word, Excel and other files)
* PDF Documents
* Images (if less than three)
* Video Recordings

**COMPRESS** the following:

* Project Code
* Images if more than 3

Any singe submission must contain all required components unless stated.

Submissions must be completed **BEFORE 5PM** on the date specified at the beginning of the assessment, unless otherwise indicated on the Learning Management System (Blackboard).

# Appendix B: Code Style and Commenting

### Code File Headers

At the start of EVERY file written in a C-style language (C, C++, C#, PHP, JavaScript, et al), the following block of comment is required, and must be completed with the appropriate information:

/\*\*

\* Assessment Title: Portfolio Part X

\* Cluster: Intermediate RIoT

\* Qualification: ICT50220 Diploma of Information Technology (Advanced Programming)

\* Name: YOUR NAME

\* Student ID: xxxxxxxxx

\* Year/Semester: 2024/S2

\*

\* YOUR SUMMARY OF PORTFOLIO ACTIVITY

\* GOES HERE

\*

\*

\* Components & Identifiers:

\* - Arduino Uno R3 UNO\_1

\* - Breadboard BBOARD\_1

\* - LED LED\_R\_1 [Red]

\* - Resistor RES\_1 [xxxxΩ]

\* - Pushbutton Switch PUSH\_1

\*

\*/

### Code Style (Naming Conventions)

For code written in a C-style language (C, C++, C#, PHP, JavaScript, et al), the following will be required…

#### Case (Upper/Lower/Mixed)

|  |  |  |
| --- | --- | --- |
| Case | Use For… | Example |
| Camel Case | Variables  Methods  Functions | ledState  toggleSwitch()  toggleLed() |
| Pascal Case | Class names | class Led() { …} |
| Snake case |  | bonus\_value |
| Shouty/Angry Snake Case | Constants | LED\_1 |

#### Length

|  |  |  |
| --- | --- | --- |
| Use | Requirements | Example |
| Variables, Constants, Methods, Functions, Class Names… | Minimum one word  NO abbreviations | ledState  LED\_1\_PIN  taxRate |

### Code Style (Formatting)

Code must be formatted consistently to facilitate ease of reading, debugging and collaboration. The following will be used as basic requirements:

|  |  |  |
| --- | --- | --- |
| Rule | Requirements | Example |
| Indenting | Multiples of 2 or 4 spaces  Do not mix indent sizes | if (switchState == LOW) {  setLedOn(LED\_1);  } |

### Code Documentation (All Comments)

Commenting of code will depend on the requirements of the assessment or project. The following are good guidelines for good code documentation for use in your work:

| Rule | Requirements | Example |
| --- | --- | --- |
| Value | Comments must add value to the code | // Calculate the power using Ohm’s law |
| Length | Lines should be less than 96 characters including prefixing symbols | // Determine taxation rate  /\*\*  \* Determine taxation rate  \*/ |

### Code Documentation (Doc Blocks)

Doc Blocks are used for commenting of methods, function, and classes. They are required to explain what the purpose is and how to use the item being described.

| Rule | Requirements | Example |
| --- | --- | --- |
| Doc Block | Provide summary details  Used for functions, methods and classes  Defines inputs and types  Defines output and types  Start with /\*\*  Each line starts with: \*  Last line: \*/  General Structure:   * First line after /\*\* is a one sentence short description * One blank line * Optional longer explanation with example usages * One blank line * Inputs * One blank line * Outputs   Inputs and Outputs are optional, so if the function/method does not contain these then the detail may be omitted. | /\*\*  \* LED On  \*  \* @input int ledPin  \*/ |

# Appendix C: Referencing

You will be expected to use MyBib (https://mybib.com) to collate and create your references.

We DO NOT expect a university style references section with in text citations.

We DO expect to see any references to use APA 6 or APA 7 style

We DO expect to see references added after answers to questions.

For example:

Imagine that you’re working on a project locally and bump into an exception. You try to figure out the problem, but you’re unable to find a solution. In that case, you might want to ask a colleague for help.

*Introducing Laravel Error Share - Blog*. (2024, June 6). Flare. <https://flareapp.io/blog/introducing-laravel-error-share>

# Appendix D: Video Recordings

Video recording may be a required part of many of your submissions.

You may be required to record whilst demonstrating components of the assessment.

The following list is a set of basic requirements for video recordings:

* Recording MUST be done in LANDSCAPE mode only (image is WIDE not tall).
* The video MUST be recorded in a SINGLE take.
* No editing permitted.
* At the start you must:
  + Show your face
  + Verbally state the Cluster name
  + Verbally state the Assessment title
  + Verbally stating your name
  + Verbally state your student number
* When demonstrating you are expected to explain what you are showing.
* At the end of demonstrating you are expected to state your name one more.

# Appendix E: File Naming

## Arduino Projects

File names for Arduino projects must be in the form:

xxx-InterRiot-ATN-PtX

Where...

* xxx is replaced by your initials
* ATN is replaced by AT1, AT2, etc.
* PtX is replaced by Pt1, Pt2, etc.

If a file is needed to be duplicated and renamed, then add -T, where T is A, B, C, D, etc.

## Git Ignore File

Download the [rename-me.gitignore](https://blackboard.northmetrotafe.wa.edu.au/bbcswebdav/pid-4387842-dt-content-rid-59526878_1/xid-59526878_1) file (available on Blackboard).

Rename it to: .gitignore

Move this file into the folder with your code repository for this portfolio **BEFORE** you add or commit any code.